粒子群算法进阶应用

- 求解方程组
- 多元函数拟合
- 拟合微分方程

粒子群算法：在给定的解空间中进行搜索，找到一个解使得某个目标函数(也可能为一个很复杂的式子)达到最大或者最小。

求解方程组

方程组的一般形式

m个方程，n个未知数的方程组可以记为：

$$\begin{cases} f_1(x_1, x_2, \cdots, x_n) = 0 \\ f_2(x_1, x_2, \cdots, x_n) = 0 \\ \vdots \\ f_m(x_1, x_2, \cdots, x_n) = 0 \end{cases}$$

思考：怎么用粒子群算法找到这个方程组的一个解？

定义适应度函数(目标函数)

m个方程，n个未知数的方程组可以记为：

$$\begin{cases} f_1(x_1,x_2,\cdots,x_n)=0 \\ f_2(x_1,x_2,\cdots,x_n)=0 \\ \vdots \\ f_m(x_1,x_2,\cdots,x_n)=0 \end{cases}$$

目标函数可以定义为：$\sum_{i=1}^{m}[f_i(x_1,x_2,\cdots,x_n)]^2$

也可以定义为$\sum_{i=1}^{m}|f_i(x_1,x_2,\cdots,x_n)|$

显然，我们要找到一组$x_1,x_2,\cdots,x_n$使得目标函数尽可能小

（理想状态下，目标函数等于0时，就找到了这个方程组的解）

一个具体的实例

求解这个方程组：

$$\begin{cases} |x_1 + x_2| - |x_3| = 0 \\ x_1 x_2 x_3 + 18 = 0 \\ x_1^2 x_2 + 3x_3 = 0 \end{cases}$$

三种方法解决这个问题：

（1）更新14讲到的vpasolve函数

（2）更新14讲到的fsolve函数

（3）粒子群算法

vpasolve函数

```
%% （1）更新14讲到的vpasolve函数
clear; clc
syms x1 x2 x3
eqn =  [abs(x1+x2)-abs(x3)  == 0, x1*x2*x3+18 == 0, x1^2*x2+3*x3 == 0]
[x1, x2, x3] = vpasolve(eqn, [x1, x2, x3], [0 0 0])
[x1, x2, x3] = vpasolve(eqn, [x1, x2, x3], [1 1 1])  % 换一个初始值试试
```

```
x1 =

Empty sym: 0-by-1

x2 =

Empty sym: 0-by-1

x3 =

Empty sym: 0-by-1
```

初始值为[0,0,0]时

```
x1 =

6.803057353234908404354 2928614427

x2 =

-0.414133803887221114709056357042 36

x3 =

6.388923549347687289645236504400 3
```

初始值为[1,1,1]时

fsolve函数

```
%% （2）更新14讲到的fsolve函数
x0 = [0,0,0];  % 初始值
x = fsolve(@my_fun,x0)
% 换一个初始值试试
x0 = [1,1,1];  % 初始值
x = fsolve(@my_fun,x0)
```

```
function F = my_fun(x)
    F(1) =  abs(x(1)+x(2))-abs(x(3));
    F(2) =  x(1) * x(2) * x(3) + 18;
    F(3) = x(1)^2*x(2)+3*x(3);
end
```

No solution found.

初始值为[0,0,0]时

```
x =

    6.8031   -0.4141    6.3889
```

初始值为[1,1,1]时

粒子群算法

```
%% (3) 粒子群算法（可以尝试多次，有可能找到多个解）
clear; clc
narvs = 3;
% 使用粒子群算法，不需要指定初始值，只需要给定一个搜索的范围
lb = -10*ones(1,3);  ub = 10*ones(1,3);
options = optimoptions('particleswarm','FunctionTolerance’   ,
1e-12,'MaxStallIterations',100,'MaxIterations',20000,'SwarmSize',100);
[x, fval] = particleswarm(@Obj_fun,narvs,lb,ub,options)
```

```
function f = Obj_fun(x)
    f1= abs(x(1)+x(2))-abs(x(3)) ;
    f2 = x(1) * x(2) * x(3) + 18;
    f3= x(1)^2 * x(2) + 3*x(3);
    f = abs(f1) + abs(f2) + abs(f3);
end
```

```
x =

    1.4206   -4.3401    2.9195

fval =

    2.2204e-15
```

```
x =

    6.8031   -0.4141    6.3889

fval =

    6.5103e-13
```

可以得到不同的解，但不是每次都能找到这么好的解，需要多次运行。

求解得到的结论

三种方法解决这个问题：

（1）更新14讲到的vpasolve函数

（2）更新14讲到的fsolve函数

（3）粒子群算法

结论：

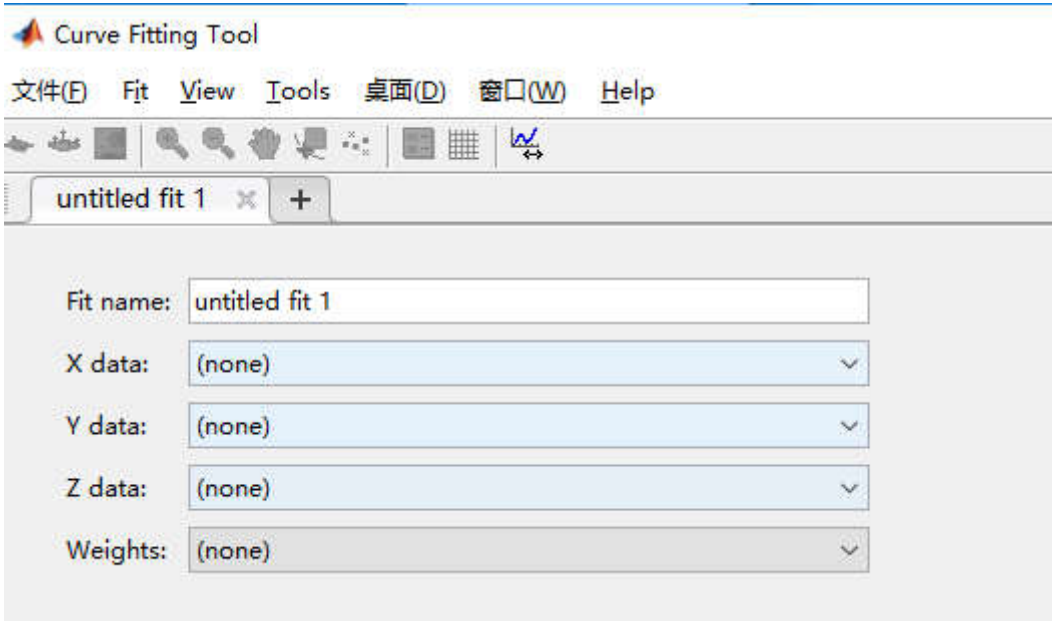
（1） vpasolve函数和fsolve函数需要给定一个比较好的初始值，如果初始值没给好则求不出结果；

（2）粒子群算法不需要给初始值，只需要给一个搜索的范围。由于算法本身具有随机性，因此可能需要多次运行才能得到一个较好的结果。

多元函数拟合

为什么现在还讲拟合?

第四讲学的拟合中，我们使用的是 Matlab自带的拟合工具箱。



然而，拟合工具箱只能对一元函数和二元函数进行拟合。

来看一个更复杂的问题（三元非线性函数）

用下表的观测数据，拟合$y=e^{-k_1x_1}\sin(k_2x_2)+x_3^2$中的未知参数$k_1$和$k_2$.

序号	y	x_1	x_2	x_3	序号	y	x_1	x_2	x_3
1	15.02	23.73	5.49	1.21	14	15.94	23.52	5.18	1.98
2	12.62	22.34	4.32	1.35	15	14.33	21.86	4.86	1.59
3	14.86	28.84	5.04	1.92	16	15.11	28.95	5.18	1.37
4	13.98	27.67	4.72	1.49	17	13.81	24.53	4.88	1.39
5	15.91	20.83	5.35	1.56	18	15.58	27.65	5.02	1.66
6	12.47	22.27	4.27	1.50	19	15.85	27.29	5.55	1.70
7	15.80	27.57	5.25	1.85	20	15.28	29.07	5.26	1.82
8	14.32	28.01	4.62	1.51	21	16.40	32.47	5.18	1.75
9	13.76	24.79	4.42	1.46	22	15.02	29.65	5.08	1.70
10	15.18	28.96	5.30	1.66	23	15.73	22.11	4.90	1.81
11	14.20	25.77	4.87	1.64	24	14.75	22.43	4.65	1.82
12	17.07	23.17	5.80	1.90	25	14.35	20.04	5.08	1.53
13	15.40	28.57	5.22	1.66					

来源：司守奎《数学建模算法与应用第二版》P95

最小二乘法求解拟合问题

假设因变量为y，自变量为$x_1,x_2,\cdots,x_k$，我们收集到了N个样本点：

$$(Y_i,X_{1i},X_{2i},\cdots,X_{ki}),\ i=1,2,\cdots,N$$

给定拟合函数$y=f(x_1,x_2,\cdots,x_k|\theta_1,\theta_2,\cdots,\theta_m)$，其中$\theta_1,\theta_2,\cdots,\theta_m$是未知的待估参数

最小二乘法的思想就是：$\hat{\theta}_1,\hat{\theta}_2,\cdots,\hat{\theta}_m=\underset{\theta_1,\theta_2,\cdots,\theta_m}{arg\min}\sum_{i=1}^{N}[Y_i-f(X_{1i},X_{2i},\cdots,X_{ki}|\theta_1,\theta_2,\cdots,\theta_m)]^2$

用文字解释就是：我们估计出来的这组参数是使得样本的残差平方和最小的那一组参数

注意：最小二乘法本质上就是一个求无约束目标函数最小值的问题

这个目标函数决策变量就是我们待估的未知参数$\theta_1,\theta_2,\cdots,\theta_m$

将最小二乘法视为最值问题

用下表的观测数据，拟合$y=e^{-k_1x_1}\sin(k_2x_2)+x_3^2$中的未知参数$k_1$和$k_2$.

序号	y	x_1	x_2	x_3
1	15.02	23.73	5.49	1.21
2	12.62	22.34	4.32	1.35
3	14.86	28.84	5.04	1.92
...	...	...	...	...
24	14.75	22.43	4.65	1.82
25	14.35	20.04	5.08	1.53

找到k_1和k_2使得下面这个函数的值最小。

$$
\begin{aligned}
&\{15.02-[e^{-23.73k_1}\sin(5.49k_2)+1.21^2]\}^2\\
+&\{12.62-[e^{-22.34k_1}\sin(4.32k_2)+1.35^2]\}^2\\
+&\{14.86-[e^{-28.84k_1}\sin(5.04k_2)+1.92^2]\}^2\\
+&\qquad\qquad\cdots\\
+&\{14.75-[e^{-22.43k_1}\sin(4.65k_2)+1.82^2]\}^2\\
+&\{14.35-[e^{-20.04k_1}\sin(5.08k_2)+1.53^2]\}^2
\end{aligned}
$$

利用fmincon函数

主函数 code1.m

```
clear; clc
global x y;  % 将x和y定义为全局变量（方便在子函数中直接调用，要先声明）
load data_x_y  % 数据集内里面有x和y两个变量
k0 = [0 0];  % 初始值
lb = []; ub = [];
[k, fval] = fmincon(@Obj_fun,k0,[],[],[],[],lb,ub)
```

子函数Obj_fun.m

```
function f = Obj_fun(k)
    global x y;  % 在子函数中使用全局变量前也需要声明下
    y_hat = exp(-k(1)*x(:,1)) .* sin(k(2)*x(:,2)) + x(:,3).^2;
    f = sum((y - y_hat) .^ 2);
end
```

fval = 552.6961
（也可以更换为其他的初始值，这时候的值可能会变化）

利用无约束最小值函数

知乎：https://www.zhihu.com/question/31052709

```
% Nelder-Mead单纯形法求解最小值，适用于解决不可导或求导复杂的函数优化问题
[k, fval] = fminsearch(@Obj_fun,k0)

% 拟牛顿法求解无约束最小值，适用于解决求导容易的函数优化问题
[k, fval] = fminunc(@Obj_fun,k0)
```

fminunc与fminsearch函数的区别是什么?

两个都是求极小值，有区别吗?

关注问题 | 写回答 | 邀请回答 | 添加评论 | 分享 | ···

1 个回答 默认排序

张国王
中科院人工智能研究人员，主攻大规模遥感图像智能化解译

17 人赞同了该回答

共同点：
算法属性：局部最优化算法
适用范围：无约束多变量最优化问题
不同点：两个函数使用的方法不一样
fminunc 采用拟牛顿法(QN)，是一种使用导数的算法。参见拟牛顿法 分析与推导
fminsearch 采用Nelder-Mead单纯形法，是一种直接搜索法。参考单纯形法、Simplex Method

发布于 2015-11-26

▲ 赞同 17 ▼ | 添加评论 | 分享 | 收藏 | 喜欢

```
fval =  297.0634
```

（也可以更换为其他的初始值，这时候的值可能会变化）

非线性最小二乘拟合函数

主函数 code2.m

```
clear; clc
load data_x_y % 数据集内里面有x和y两个变量
k0 = [0 0]; % 初始值
lb = []; ub = [];
[k, fval] = lsqcurvefit(@fit_fun,k0,x,y,lb,ub) % 使用lsqcurvefit函数
```

子函数fit_fun.m

```
function y_hat = fit_fun(k,x)
    y_hat = exp(-k(1)*x(:,1)) .* sin(k(2)*x(:,2)) + x(:,3).^2;
End
```

edit lsqcurvefit

LSQCURVEFIT attempts to solve problems of the form:
% min sum {(FUN(X,XDATA)-YDATA).^2}
X = LSQCURVEFIT(FUN,X0,XDATA,YDATA) starts at X0 and finds coefficients X to best fit the nonlinear functions in FUN to the data YDATA (in the least-squares sense).
FUN accepts inputs X and XDATA and returns a vector (or matrix) of function values F, where F is the same size as YDATA, evaluated at X and XDATA.

fval = 297.0634
（也可以更换为其他的初始值，这时候的值可能会变化）

使用粒子群算法

主函数 code3.m

```
clear; clc
global x y;  % 将x和y定义为全局变量（方便在子函数中直接调用，要先声明）
load data_x_y  % 数据集内里面有x和y两个变量
narvs = 2;
% 使用粒子群算法，不需要指定初始值，只需要给定一个搜索的范围
lb = [-10 -10];  ub = [10 10];
[k,fval] = particleswarm(@Obj_fun,narvs,lb,ub)
```

子函数Obj_fun.m

```
function f = Obj_fun(k)
    global x y;  % 在子函数中使用全局变量前也需要声明下
    y_hat = exp(-k(1)*x(:,1)) .* sin(k(2)*x(:,2)) + x(:,3).^2;
    f = sum((y - y_hat) .^ 2);
end
```

使用粒子群算法的优点：只需要给定一个搜索的范围，不需要指定初始值。
（大家可以用前面的方法尝试，找到一个合适的初始值真的很难ヽ(ﾟ▽ﾟ)ﾉ）

```
fval =  297.0634
```

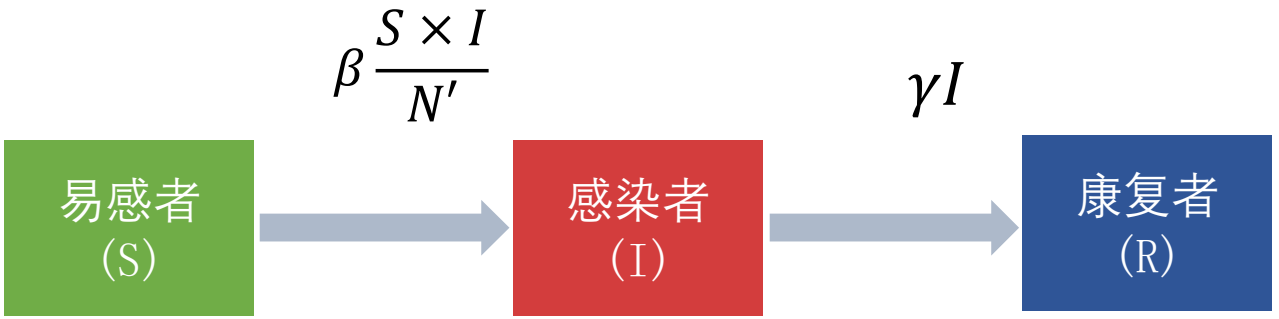
（粒子群算法可以运行多次，从中选择最好的结果）

粒子群算法进阶应用3

拟合微分方程

注意：如果微分方程或者微分方程组有解析解（dsolve函数能求出来），那么这时候就转换为了函数拟合问题，例如我们在更新15里面举的美国人口预测问题(阻滞增长模型)；我们这里讨论的拟合微分方程指的是只能用ode45这类函数求出数值解的情况。（详见：更新15微分方程）

回顾SIR模型(更新15)



对应的微分方程组为：

$$\begin{cases} \frac{dS}{dt} = -\beta\frac{S\times I}{N'} \\ \frac{dI}{dt} = \beta\frac{S\times I}{N'} - \gamma I \\ \frac{dR}{dt} = \gamma I \end{cases}, \text{且} N' = S + I$$

β：易感染者与已感染者接触且被传染的强度， γ：康复率

注意，这里的分母我们取的是N'，其表示的含义为传染病系统中的有效人群，即不包含脱离了系统的人群(例如：获得了抗体、死亡或者被有效隔离的人)。

典型例题

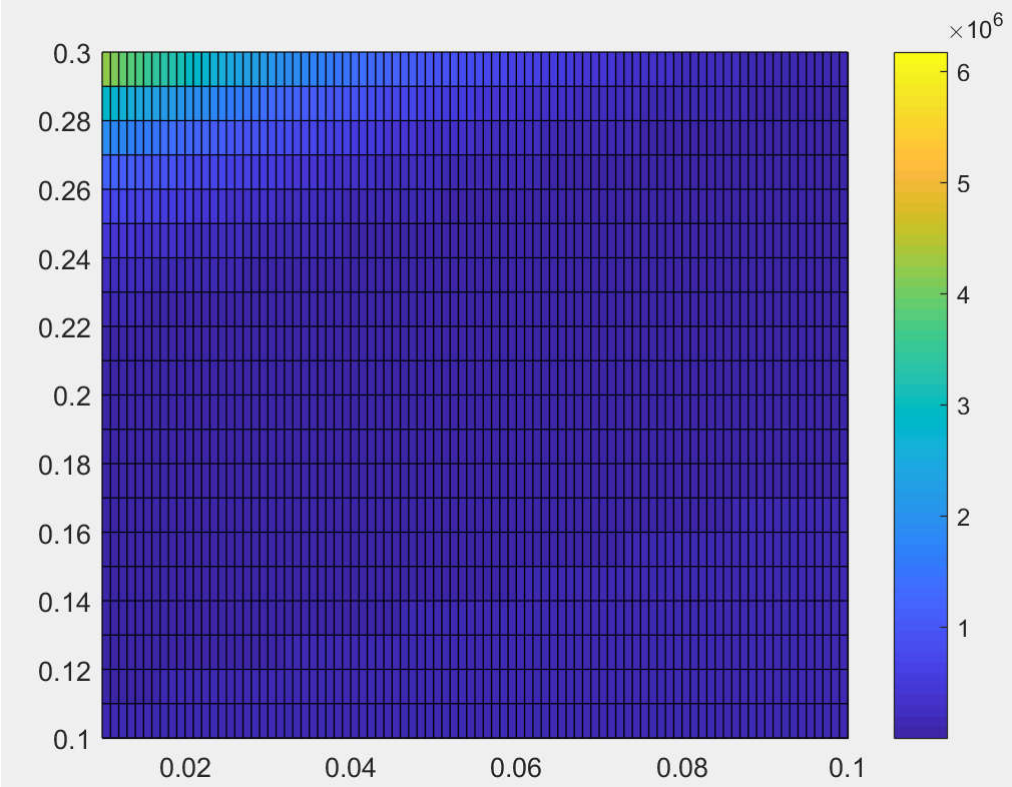
某区域共有10000人（假设不考虑人口的生死以及迁入迁出），现在某科研团队收集了某疾病暴发前20天的各类人口数据，请建立SIR模型，并用下面的数据去拟合SIR模型中的两个未知参数:β和γ，并预测未来7天该疾病的变化趋势。

时间	S	I	R	时间	S	I	R
第1天	9994	6	0	第11天	9956	38	6
第2天	9992	8	0	第12天	9951	40	9
第3天	9990	10	0	第13天	9939	51	10
第4天	9988	11	1	第14天	9926	62	12
第5天	9985	13	2	第15天	9907	79	14
第6天	9980	17	3	第16天	9894	90	16
第7天	9979	18	3	第17天	9885	96	19
第8天	9974	22	4	第18天	9878	101	21
第9天	9969	26	5	第19天	9858	118	24
第10天	9964	31	5	第20天	9841	132	27

注意：本数据不是真实数据，仅用于讲解需要。

网格搜索

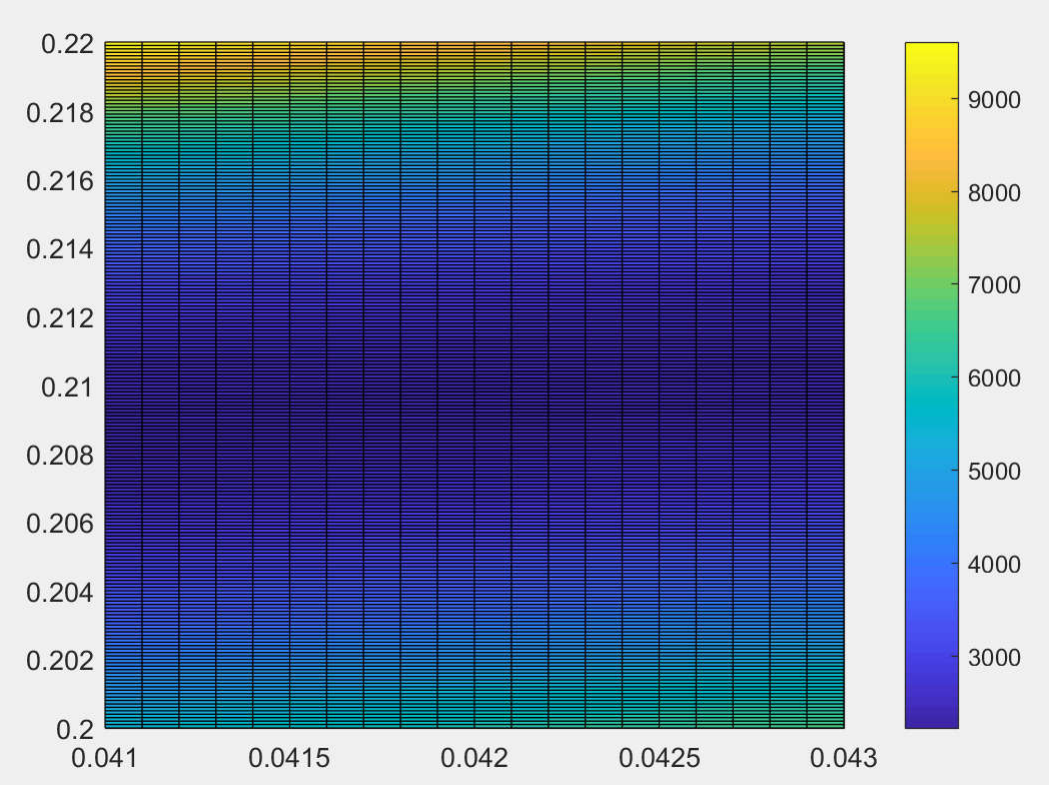
BETA = 0.1:0.01:0.3; % 估计一个BETA所在的范围
GAMMA = 0.01:0.001:0.1; % 估计一个GAMMA所在的范围



% beta =0.2100
% gamma =0.0420
% sse =2302

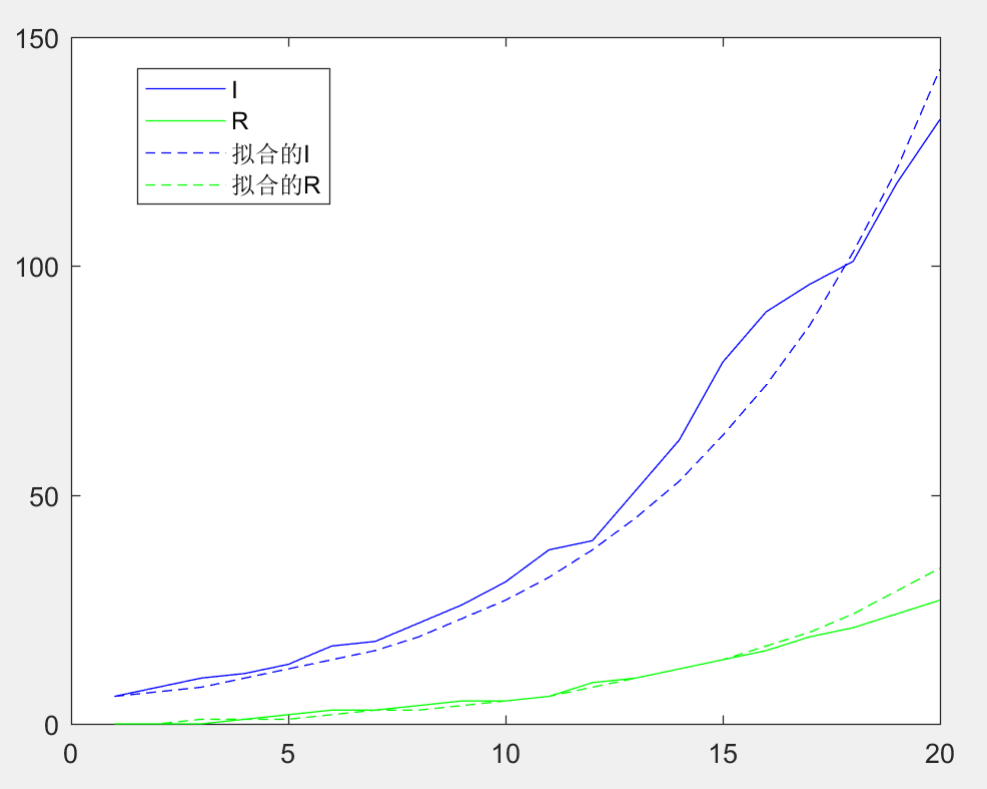
缩小搜索范围

```
BETA = 0.2:0.0001:0.22; % 缩小BETA所在的搜索范围
GAMMA = 0.041:0.0001:0.043; % 缩小GAMMA所在的搜索范围
```




```
% beta =0.2094
% gamma =0.0415
% sse =2219
```

网格搜索法



网格搜索法实际上就是枚举法，下面是对它的总结：

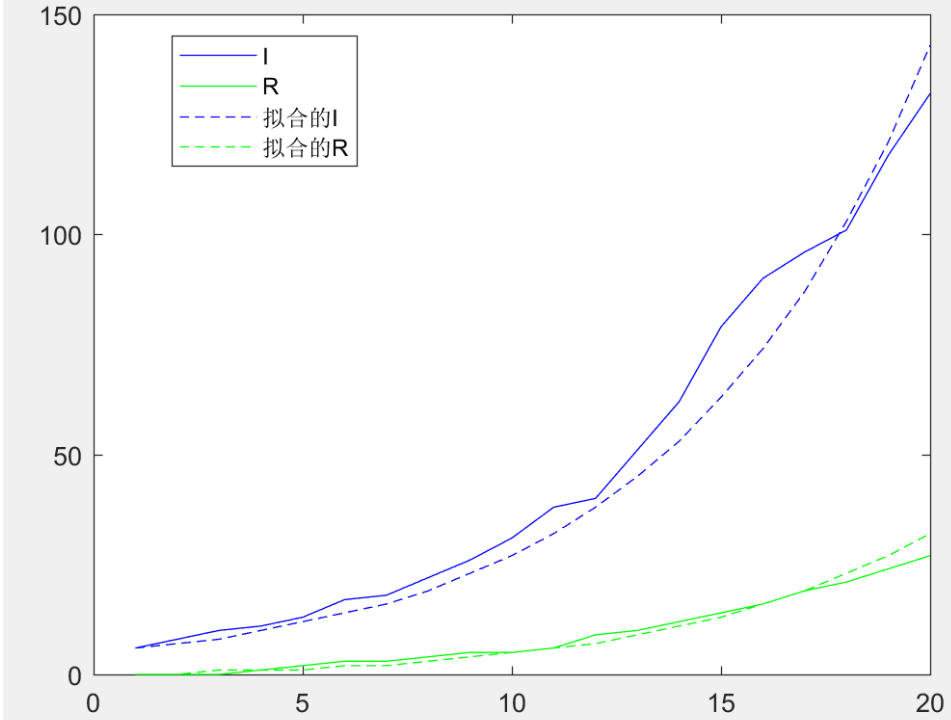
（1）搜索的精度越高（网格划分的越细），搜索耗费的时间越长；

（2）缩小网格搜索范围（可以减少搜索时间）可能会让我们找到的解陷入局部最优；

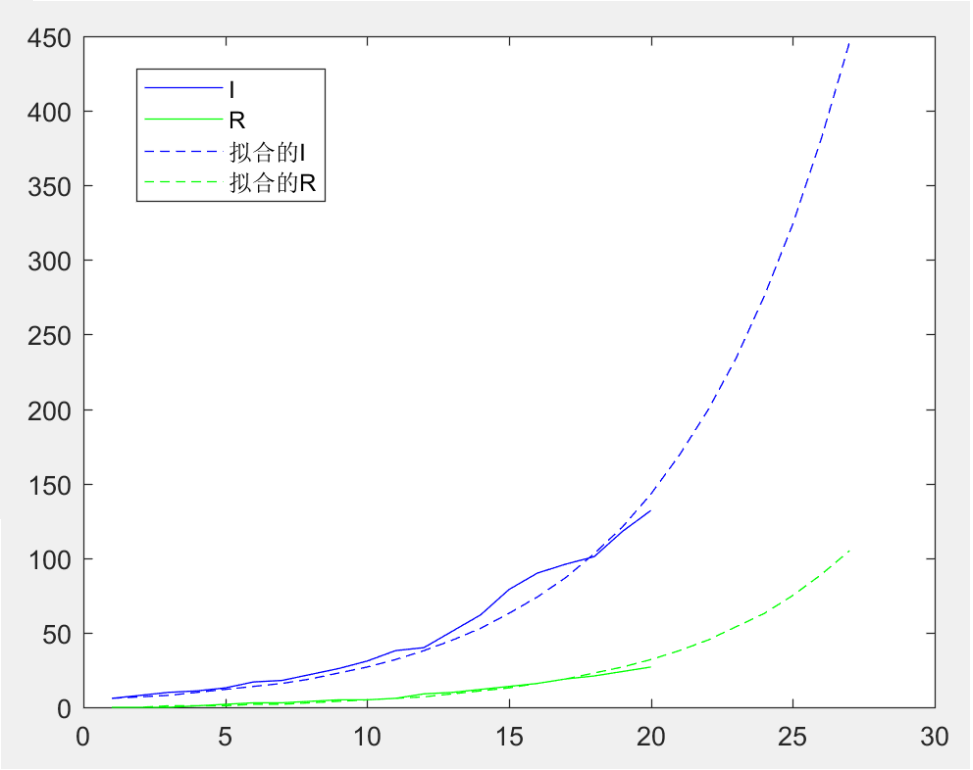
（3）如果我们有多个要搜索的变量，网格搜索法就很难办了，多重循环会大大增加搜索时间。

```
% beta =0.2094
% gamma =0.0415
% sse =2219
```

粒子群算法运行的效果图



注意：粒子群算法计算的结果具有随机性，结果可能会变换，大家可以多次运行取一个最好的结果。




```
% beta =0.2067
% gamma =0.0388
% sse =2185
```

高手的玩法：参数动起来

对应的微分方程组为：

$$\begin{cases} \dfrac{dS}{dt} = -\beta \dfrac{S \times I}{N'} \\ \dfrac{dI}{dt} = \beta \dfrac{S \times I}{N'} - \gamma I \ ，且 N' = S + I \\ \dfrac{dR}{dt} = \gamma I \end{cases}$$

β：易感染者与已感染者接触且被传染的强度， γ：康复率

如果β和γ都和时间t有关该怎么去拟合？

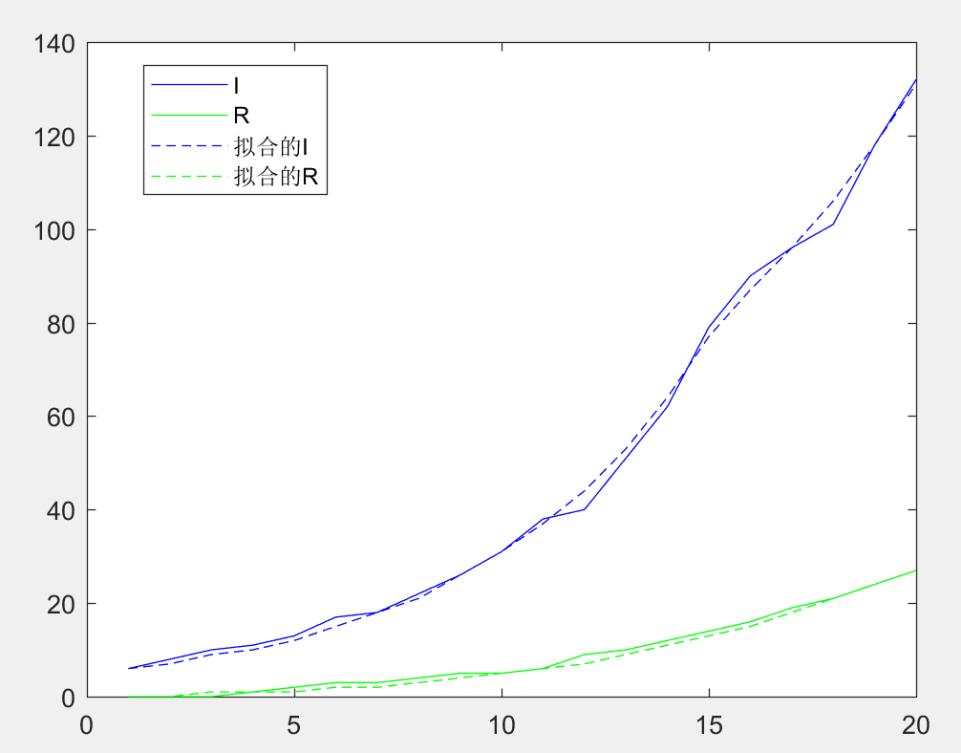
举个例子：

$$\beta = \begin{cases} \beta_1, \ t \le 15 \\ \beta_2, \ t > 15 \end{cases} , \gamma = a + bt$$

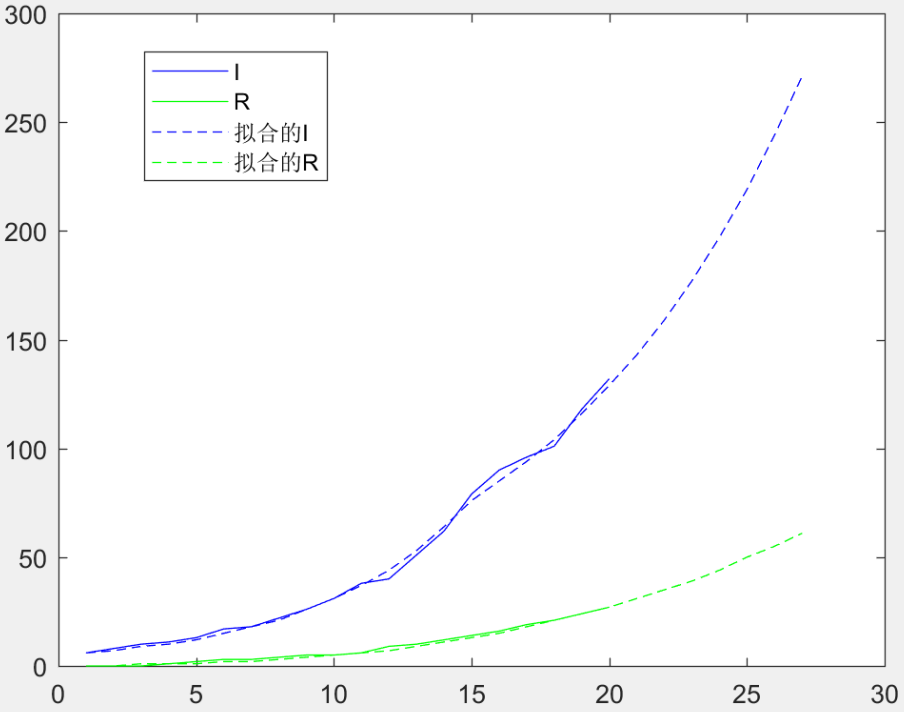
那么怎么去拟合β_1, β_2, a以及b呢？

（这是我随便给的例子，大家实际做题时要根据你的数据来设计这个函数）

参数动起来后的效果图



注意：粒子群算法计算的结果具有随机性，结果可能会变换，大家可以多次运行取一个最好的结果。




```
% beta1 =0.2238
% beta2 =0.1206
% a =0.0518
% b =-0.0015
% sse =153
```